

rnabob, an RNA pattern searcher: user's guide

What rnabob does

RNABOB allows searching a sequence database for RNA structural motifs. The probe motif is specified in a “descriptor” file, which describes its primary sequence, secondary structure, and tertiary constraints.

Running rnabob

To run RNABOB, type:

```
rnabob descriptor-file seqfile
```

where `descriptor-file` and `seqfile` are the names of the file which contains the descriptor you want to search for and the name of the sequence file you want to search.

Sequence database format

`rnabob` can read sequence files in most any common format. FASTA, GCG, EMBL, GenBank, and other formats are acceptable. If you type your own sequence in, the easiest format is FASTA format. In FASTA format, a `>` precedes a one-word name for the sequence, followed by a free format description line; subsequent lines contain the sequence itself.

```
>CHKHSPA (GenBank) Chicken 108K heat shock gene, complete cds.  
AACGCATCAGACTACGACTAC  
CAGTACGCTGCTGCTGACTGC
```

A FASTA file can contain any number of sequences, each preceded by a name/description line with a `>` as the first character.

If the environment variable `BLASTDB` is set (which it will tend to be if you have BLAST software installed on your system), `rnabob` will look in that directory as well as the current directory when it tries to find `seqfile`. (The current directory is searched first.) This lets you save some typing if you're searching one of the main sequences databases on your system – you don't have to type the full path to the database.

It is advisable to construct small databases for testing new descriptors.

Descriptor file syntax

The descriptor file syntax is fairly powerful, and allows a great deal of freedom for specifying RNA motifs. The syntax is therefore a bit complicated. Check out the files `trna.des`, `r17.des`, and `pseudoknot.des` for example descriptors if you get lost in this information. (They are included with the `rnabob` source distribution.)

The descriptor file has two parts: a “topology” description and an “explicit” description.

The first non-blank, non-comment line of the file is the topology description. It defines the order of occurrence of a series of single-stranded, double-stranded, and related elements. Each element must be given a unique name (a number, typically) and must be prefixed with 's', 'h', or 'r', indicating single-strand, helical, or a relational element. Helical and relational elements are paired to other elements, which are suffixed by a prime, '.

For example:

```
h1 s1 h1'
```

describes a hairpin loop structure with a simple helix and single-stranded loop. If the helix always contained a non-canonical base pair at one position, the topology could be described as:

```
h1 r1 h2 s1 h2' r1' h1'
```

where r1, r1' indicate a correlation, where the sequence of r1 constrains the sequence of r1'. (Helices are a special case of this.)

The remaining non-comment, non-blank lines are explicit descriptions of each element in turn. Each line contains 3 or 4 fields, separated by tabs or blank space. The first field is the name of the element, from the topology description. The second field is the number of mismatches allowed in this element. The third field is the primary sequence constraint to apply to this element.

Helices and relational element pairs are specified on a single line rather than two. Mismatches and primary sequence constraints are given as pairs, separated by a colon “:”. The left side is the constraint applied to the upstream element, and the right side is applied to the downstream elements.

The primary sequence constraint is given as a sequence of nucleotides. Any IUPAC single-letter code is recognized, including N if the position can have any base identity. Allowed length variations are specified with asterisks *, where each * will allow either 0 or 1 N at that position. For example,

```
GGAGG*****NNNAUG
```

specifies a GGAGG Shine/Dalgarno site and an AUG initiation codon, separated by a spacer of 3 to 9 nucleotides of any sequence.

An alternative syntax can be used for very long gaps. GGAGG[10]NNNAUG is the same as GGAGG*****NNNAUG.

Be careful defining variable length helices and relational elements; if the number and type (gap or identity) of position do not match on left and right sides, the program will refuse to accept the descriptor.

Relational elements have an additional field which specifies a “transformation matrix” of four nucleotides, specifying the rule for making the r' pattern from the r sequence in order A-C-G-T. For example, the transformation matrix for a simple helix is TGCA; if you allow G-U pairs, it is TGYR. RNABOB allows G-U pairing by default and uses the TGYR matrix for helical elements.

For example, the explicit description of our hairpin might be:

```
h1 0:0 NNN:NNN
```

```
r1 0:0 R:N GNAN
h2 0:0 **NC:GN**
s1 0 UUCG
```

This describes a stem of 6 to 8 base pairs, in which the 4th pair from the bottom of the stem must be a non-canonical GA pair. Note that, in general, the left side of the primary constraint for helices and relational elements is redundant, and should be given as all N. In some cases it is convenient to constrain the right side to require a particular base pair (GU, for instance) at one position.

A note on mismatches: The split format for helices and relational elements works like this. The number on the left constrains the primary sequence match of the left side of the primary constraint. The number on the right constrains the match of the right side of the primary constraint, *after* that side has been constructed according to the sequence on the left. In other words, the number on the left constrains the mismatches in primary sequence only, while the number on the right will constrain the number of mispaired positions in the helix.

Finally: any line that begins with a pound sign # is a comment line, and will not be interpreted by the pattern compiler.

Examples

Example descriptors in this release include `trna.des` (a general descriptor of a tRNA structure), `r17.des` (descriptor of the consensus binding site for the r17 phage coat protein), and `pseudoknot.des` (descriptor of a simple pseudoknotted structure).

An example cosmid `F22B7.fa` from the *C. elegans* genome sequencing project is also provided for running these descriptors against. `trna.des` hits twice, once on each strand. (F22B7 has several other tRNA genes in it which the pattern fails to detect – this is *not* a pattern to use for tRNA genefinding!) `r17.des` hits once – not that R17 coat protein ever sees *C. elegans* sequence.

For instance,

```
rnabob trna.des F22B7.fa
```

searches the top strand of the cosmid. (To search both strands, use the `-c` option; see below in the Options section.) The output will appear on the screen as the program runs. To save it in a file, redirect the output when you give the command:

```
rnabob trna.des F22B7.fa > rnabob.out
```

If you run something against the full database (which will take at least 10 minutes and possibly MUCH longer, depending on the degeneracy of the descriptor) or you're running things in the middle of the day, you can be polite to the system and other users and downgrade the priority of your run:

```
nice rnabob trna.des F22B7.fa > rnabob.out
```

This usually has almost no effect on your run time but improves system performance for everyone else, so it's a good thing to do.

If you want to walk away from the terminal (running `rnabob` in the background), add a `&` to the command:

```
nice rnabob trna.des F22B7.fa > rnabob.out &
```

Options

The behavior of `rnabob` can be modified with certain command line options.

- c Complement: search for the pattern on the complementary strands too.
- h Help: print out some brief usage and version info for the program.
- q Quiet: suppress the verbose header on the output. Useful for directly piping `rnabob` output into another program or filter.
- s Skip: a hack to avoid a problem in the DNABANK. There are some sequences in the database which have long stretches of ambiguous sequence (N's). Descriptors with no primary sequence constraints will match these garbage sequences at many, many positions, and generate huge outputs (sometimes enough to crash the program and Beagle). `SKIPMULT` toggles a search strategy that skips forward a pattern-length rather than a single base when a match is found, thus printing out only a single match when overlapping matches are found.
- F Fancier output: print out the sequence that matched. Still pretty rudimentary.

References

Gautheret D., Major F., and Cedergren R. Pattern searching/alignment with RNA primary and secondary structures: an effective descriptor for tRNA. *CABIOS* **6**:325–331, 1990.

Eddy S.R. RNABOB: a program to search for RNA secondary structure motifs in sequence databases. Unpublished.